Knowledge-Based Systems 16 (2003) 129–136

Knowledge-Based SYSTEMS

www.elsevier.com/locate/knosys

Combining symbolic classifiers from multiple inducers

José Augusto Baranauskas*, Maria Carolina Monard

Laboratory of Computational Intelligence, Department of Computer Science and Statistics, Institute of Mathematics and Computer Sciences, University of São Paulo, Av. Trabalhador Socarlense, 400 P.O. Box 668, Sao Carlos, Sao Paulo 13566-590, Brazil

Received 8 August 2001; accepted 4 February 2002

Abstract

Classification algorithms for large databases have many practical applications in data mining. Whenever a dataset is too large for a particular learning algorithm to be applied, sampling can be used to scale up classifiers to massive datasets. One general approach associated with sampling is the construction of ensembles. Although benefits in accuracy can be obtained from the use of ensembles, one problem is their interpretability. This has motivated our work on trying to use the benefits of combining symbolic classifiers, while still keeping the symbolic component in the learning system. This idea has been implemented in the XRULER system. We describe the XRULER system, as well as experiments performed to evaluate it on 10 datasets. The results show that it is possible to combine symbolic classifiers into a final symbolic classifier with increase in the accuracy and decrease in the number of final rules.

Keywords: Combining classifiers; Machine Learning; Data Mining

1. Introduction

Usual approaches for developing a knowledge base involve a manual formalization of expert's knowledge and encoding it in appropriate data structures. One important application of Machine Learning (ML) concerns the (semi) automatic construction of knowledge bases through inductive inference. Indeed, ML can provide an improvement of current techniques and a basis for developing alternative knowledge acquisition approaches.

One challenge in predictive Data Mining (DM) is to exploit and combine existing ML algorithms effectively. However, not all ML algorithms are designed to deal with big data, an important issue in DM.

One solution to this problem is to take several samples from the database inducing knowledge using these samples as input to any ML algorithm. After that, it is possible to use the ensemble approach to classify new instances [5,9]. Although benefits in accuracy can be obtained from the use of ensembles, one problem is their interpretability, since combining symbolic classifiers into one single classifier by a majority (or other) vote mechanism does not result into a symbolic classifier, i.e. an ensemble of symbolic classifiers is no longer symbolic. In this context, a symbolic classifier is the one that can be transformed into a set of rules.

In our work, we are interested in interpretability, i.e. our main concern is not only related to the correct classification of new instances, but also with explaining the user the reasons of classifying a new instance into one of the possible classes. In other words, we are interested in symbolic classifiers, specifically in classifiers that can be transformed into a set of rules.

In this work, we propose and describe the XRULER system that can be used to combine symbolic knowledge extracted from different samples using different inducers. The main idea is to transform each symbolic classifier into a standard *if–then* rule format before trying to merge the classifiers into a final one. These ideas are currently being implemented in the DISCOVER project, a under development computational environment designed as a generic knowledge discovery tool for DM research.

This work is organized as follows. Section 2 introduces the notation and definitions used in the remaining text. Section 3 provides a description of the XRULER system, including the basic cover algorithm, criteria for choosing the best rule, as well as classifying new instances. Section 4 outlines the experimental setup used to evaluate the XRULER system and Section 5 reports the experimental results and discussion. Finally, Section 6 presents some concluding remarks.

* Corresponding author.

E-mail addresses: augusto@fmrp.usp.br (J.A. Baranauskas); mcmonard@icmc.usp.br (M.C. Monard).

Table 1
Contingency matrix for the rule $L \rightarrow R$

	L	$\bar{L}$	
R	lr	$\bar{l}r$	$r = lr + \bar{l}r$
$\bar{R}$	$l\bar{r}$	$\bar{l}\bar{r}$	$\bar{r} = l\bar{r} + \bar{l}\bar{r}$
	$l = lr + l\bar{r}$	$\bar{l} = \bar{l}r + l\bar{r}$	$n = l + \bar{l} = r + \bar{r}$

2. Definitions and notation

A *dataset* T is a set of n classified (labeled) instances, each of them containing m features. A row i refers to the ith instance ($i = 1, 2, \ldots, n$) and column entries x_{ij} refer to the value of the jth ($j = 1, 2, \ldots, m$) feature X_j of instance i. As can be seen, instances are tuples $T_i = (x_{i1}, x_{i2}, \ldots, x_{im}, y_i) = (\vec{x}_i, y_i)$ also referred as (x_i, y_i) being implicitly assumed that x_i is a vector. The last column, $y_i = f(x_i)$, is what we try to predict given the x_i features. Each x_i is an element of the set $\mathtt{dom}(X_1) \times \mathtt{dom}(X_2) \times \cdots \times \mathtt{dom}(X_m)$, where $\mathtt{dom}(X_j)$ is the domain of the X_j feature and y_i belongs to one of the k classes, i.e. $y_i \in \{C_1, C_2, \ldots, C_k\}$.

A *complex* is a disjunction of conjunctions of feature tests in the form: 'X_i *op Value*', where X_i is a feature, *op* is an operator in the set $\{ =, \neq, <, \leq, >, \geq \}$ and *Value* is a valid feature X_i constant value, according to feature domain.

A *rule* assumes the form '**if** L **then** R' or symbolically $L \rightarrow R$, where the left rule part L and the right rule part R are both complexes without features in common, i.e. $\mathtt{features}(L) \cap \mathtt{features}(R) = \emptyset$. The left rule part is also denominated rule *condition* and the right rule part R is denominated rule *conclusion*.

A *classification rule* assumes the strict rule form '**if** L **then** class $= C_i$', where C_i belongs to the set of possible k classes $\{C_1, C_2, \ldots, C_k\}$. For simplicity, from now on we always refer to a *classification rule* as a *rule*.

The *coverage* of a rule is defined as follows. Considering the rule $L \rightarrow R$, instances that satisfy the L part (i.e. L is true for these instances) compose its *covered set*, in other words, those instances are *covered* by the rule. Instances that satisfy both L and R are *correctly covered* by the rule. Instances satisfying L but not R are *incorrectly covered* by the rule. On the other hand, instances that do not satisfy the L part are *not covered* by the rule.

Given a set of training instances, the inducer outputs a *classifier* (also called a *hypothesis*) such that, given a new instance, it accurately predicts its class label. All classifiers use stored data structures that are then interpreted as a mapping for an unclassified instance to a label [13]. Formally, in supervised classification an instance is a pair $(x_i, f(x_i))$, where x_i is the input and $f(x_i)$ is the output. The task of an inducer is, given a set of instances, to induce a function h that approximates f, since f is frequently unknown. In this case, h is called an *hypothesis* over the true function f.

A *symbolic classifier* is an hypothesis, whose description language can be transformed into a set of rules. Given a rule and a dataset, one possible way to measure the performance of that rule in the dataset is by computing its *contingency matrix*, shown in Table 1 [10]. In this table, L denotes the set of instances, where the rule condition is true and its complement, $\bar{L}$, denotes the set of instances, where the rule condition is false and analogously for R e $\bar{R}$. LR denotes the set of instances $L \cap R$, where both L and R are true, $L\bar{R}$ denotes the set of instances $L \cap \bar{R}$, where L is true, but R is false and so on. For simplicity, we denote the cardinality of the set A as a, in other words, $a = |A|$. Therefore, l denotes the number of instances in the set L, i.e. $l = |L|$, r denotes the number of instances in the set R, i.e. $r = |R|$, lr denotes the number of instances in the set LR and so on. As before, n is the total number of instances in the dataset.

The relative frequency $|A|/n = a/n$ associated to the subset A is denoted by $p(A)$, where A is a subset from the n instances. From this point of view, the relative frequency is used as a probability estimate. The notation $p(A|B)$ follows its usual definition given by Eq. (1), where A and B are both subset from the n instances set.

$$p(A|B) = \frac{p(AB)}{p(B)} = \frac{\frac{|AB|}{n}}{\frac{|B|}{n}} = \frac{\frac{ab}{n}}{\frac{b}{n}} = \frac{ab}{b} \tag{1}$$

An *ensemble* consists of a set of individual classifiers, whose predictions are combined, when predicting novel instance labels. Previous research has shown that an ensemble is often more accurate than any of the individual classifier in the ensemble, i.e. multiple classifiers have been shown to lead to improved predictive accuracy, when classifying instances that are not among the training set.

Although there are benefits in accuracy that can be obtained from the use of ensembles, since they usually reduce the error by reducing bias and variance [3,5,7], one problem is its interpretability by humans. It can be noted that combining symbolic classifiers into one single classifier by a majority (or other) vote mechanism does not result into a symbolic classifier any more, i.e. an ensemble of symbolic classifiers is no longer symbolic. Besides that, in general, ensembles can be very large. On experiments described in Ref. [11], the ensemble was 80–160 times greater than any of the individual classifier.

Another point to be noted is the possible redundancy present on the ensemble. To understand this, consider an ensemble composed by three decisions trees $\{h_1, h_2, h_3\}$. It is possible that a given branch on the tree h_1 to be exactly the same, or very similar, to another one on h_2 ou h_3. According to Dietterich and Kong [7], "Some method is need for converting a combination of trees (or other complex hypotheses) into a smaller, equivalent hypothesis. These tree are very redundant; how can we remove this redundancy while still reducing bias and variance?"

These three aspects—interpretability, size, and redundancy—bring undesirable consequences to the ensembles from the point of view of human comprehension. Therefore,

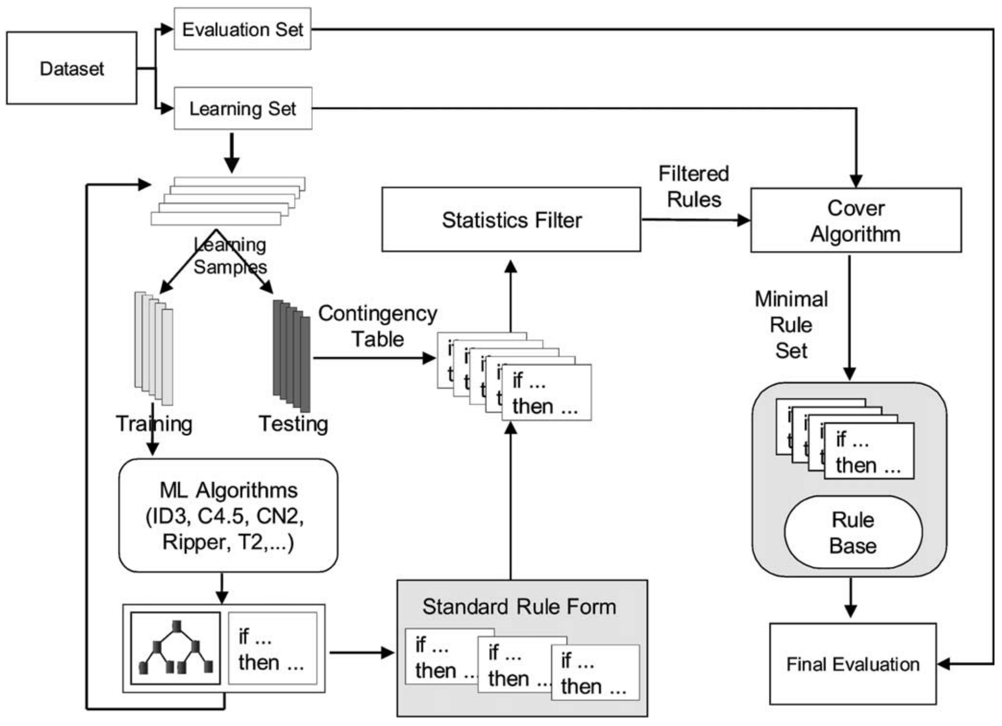

Fig. 1. The XRULER system.

alternative methods are necessary, even reducing bias and variance, in order to keep the symbolic component in the final classifier.

3. The XRULER system

One possible solution to the problem presented in Section 2 considers the combination of symbolic classifiers into a final classifier, which is still symbolic, differently of ensembles. This means that the classifiers are seen as white-boxes or, more precisely, as a rule set.

A similar idea was used in the SKICAT project proposed in Ref. [8] using the RULER system. RULER splits the data into training and testing samples. The training sample feeds the O-BTREE inducer, building a non-pruned decision tree, which is then transformed into a rule set. Since this process generates a large number of rules, RULER discards those rules that result from a weak correlation with the data. Finally, a greedy cover algorithm is applied in order to find a minimal rule set, that is, consistent with the data. According to Fayyad et al. [8], RULER produces a rule set, that is, robust and containing less rules than any of the individual tree induced by O-BTREE and it also presents a better classification performance than human experts.

In this work, we propose the extension of the RULER system to a new system named eXtended RULER (XRULER). This extension considers the use of more than one inducer in the initial learning steps, as well as different criteria that can be used by the cover algorithm.

Fig. 1 shows the XRULER system, which has been implemented using MYSQL [14] and PERL [18]. Initially, the instances are splitted into two disjoint sets: the learning set and the evaluation set. Using the learning set, XRULER extracts training and test samples. Training samples are given to inducers that extract symbolic classifiers. Each classifier is then converted into a standard *if–then* rule form [1,2] and evaluated in the correspondent testing set, computing the contingency table for each rule. The standard rule form is similar to the CN2 inducer with some extensions [15]. After that, filters can be applied in order to remove rules statistically insignificant. Then, the cover algorithm uses the learning set to find a minimal subset of rules that covers. Finally, this minimal subset is evaluated using the evaluation set, which was never seen by XRULER in the learning step.

Algorithm 1
Basic cover algorithm to select unordered rules

```
Require: InstanceSet: instances to be covered
  RuleSet: rule set
Ensure: CoverSet: minimal rule set from RuleSet that covers InstanceSet
1: procedure basic_cover(InstanceSet,RuleSet)
2: CoverSet := $\emptyset$
3: repeat
4:    best_rule := select best rule from RuleSet using InstanceSet
5:    CoverSet := CoverSet + {best_rule}
6:    RuleSet := RuleSet − {best_rule}
7:    InstanceSet := InstanceSet − {$I$: $I$ in InstanceSet and covered($I$, best_rule)}
8: until (InstanceSet = $\emptyset$) or not found (best_rule)
9: return CoverSet
```

3.1. Basic cover algorithm

The XRULER uses a cover algorithm in order to obtain a minimal rule set from the sets induced in the initial step by the symbolic ML algorithms. The cover algorithm produces unordered rules, since they are more easily understood by humans [6].

Algorithm 1 shows the basic procedure to select unordered rules. Given the rule set *RuleSet* and a set *InstanceSet* of instances to be covered, the algorithm search for the best rule *best_rule*, according to some criteria. Once the best rule is found, it is included in the cover set *CoverSet* and removed from *RuleSet*. Next, instances correctly covered by the best rule are removed from the *InstanceSet*. The reason to keep those instances incorrectly covered (removing only instances correctly covered by the best rule) allows the algorithm to find in the following iterations, a rule that correctly covers them. The function *covered*(I, *best_rule*) is true if instance I is correctly covered by the rule *best_rule*, false otherwise. This process is repeated, while there are uncovered instances or a new *best_rule* is not found.

3.2. Criteria for selecting the best rule

Algorithm 1 shows that the final result is dependent of the criteria used to select the best rule. In fact, there are several criteria and each of them provides a different minimal rule sets.

One possible criteria to select the best rule consists in choosing the rule with the best *rule fitness* for the task at hand. Denoting the degree of fitness for rule $L \rightarrow R$ as $\texttt{rf}(L \rightarrow R)$, in general, the selection criteria chooses the best rule as being the one with the greatest value of $\texttt{rf}$. It follows a description of three criteria that can be used to estimate rule fitness.

Positive reliability. It is possible to define the rule fitness $\texttt{rf}$ through its *accuracy*, where the accuracy $\texttt{acc}(L \rightarrow R)$ is defined by Eq. (2) as the conditional probability of the rule conclusion R given the fact the rule condition L is true [17].

$$\texttt{rf}(L \rightarrow R) \overset{\text{def}}{=} \texttt{acc}(L \rightarrow R) = p(R|L) \tag{2}$$

A simple estimate for $p(R|L)$ uses the positive reliability $\texttt{prel}(L \rightarrow R)$, defined as the ratio between the number of instances correctly covered by the rule and the total number of instances covered by the rule, i.e. lr/l, according to the notation introduced in Table 1.

Equivalently, $\texttt{prel}(L \rightarrow R) = lr/l$ can be expressed as $lr/(lr + l\bar{r})$, where lr is the number of instances correctly covered by the rule and $l\bar{r}$ is the number of instances incorrectly covered by the rule. It is possible to note that $0 \leq \texttt{prel}(L \rightarrow R) \leq 1$ and, according to this criteria, the larger the positive reliability, the best is the rule.

However, the positive reliability presents an undesirable property. For instance, considering two rules $R_1 \equiv L_1 \rightarrow R_1$ and $R_2 \equiv L_2 \rightarrow R_2$, with $l_1r_1 = 100$, $l_1\bar{r}_1 = 1$ and $l_2r_2 = 5$, $l_2\bar{r}_2 = 0$, then $\texttt{prel}(R_1) = l_1r_1/l_1 = 0.99$ e $\texttt{prel}(R_2) = l_2r_2/l_2 = 1.00$, indicating that R_2 is a better rule than R_1, which is not true in this situation.

Laplace's accuracy. One possible solution for the problem presented by the positive reliability, when estimating $p(R|L)$ consists in replacing it by the Laplace's accuracy ($\texttt{lacc}$) [6], defined by Eq. (3), where k is the number of classes in the dataset.

$$\texttt{rf}(L \rightarrow R) \overset{\text{def}}{=} \texttt{lacc}(L \rightarrow R) = \frac{lr + 1}{l + k} = \frac{lr + 1}{lr + l\bar{r} + k} \tag{3}$$

Considering this criteria, the larger the value of Laplace's accuracy, the best is the rule. Still considering the previous example for $k = 2$ classes, we obtain $\texttt{lacc}(R_1) = 0.98$ and $\texttt{lacc}(R_2) = 0.85$, indicating that rule R_1 is better than rule R_2. Similar to the positive reliability, the Laplace's accuracy assumes values in the range $0 \leq \texttt{lacc}(L \rightarrow R) \leq 1$.

Novelty. The novelty $\texttt{nov}(L \rightarrow R)$, defined by Eq. (4), compares the observed lr value against the expected value under the assumption of independency lr/n, where n is the number of instances [10]. Therefore, it is possible to estimate rule fitness using novelty.

$$\texttt{rf}(L \rightarrow R) \overset{\text{def}}{=} \texttt{nov}(L \rightarrow R) = p(L)(p(R|L) - p(R)) \tag{4}$$

The novelty assumes values in the range $-0.25 \leq \texttt{nov}(L \rightarrow R) \leq 0.25$ and, considering this criteria, the larger its positive value (near 0.25), the best is the rule. The expression $p(R|L)$ in Eq. (4) can be estimated using the Laplace's accuracy.

3.3. Classifying new instances

Once obtained the minimal rule set through the cover algorithm, it is possible to use these rules as a classifier. XRULER uses a similar criteria than the CN2 inducer, when labeling new instances [6]. Initially, for each rule $L \rightarrow R$, the values lr and $l\bar{r}$ are calculated using the same *InstanceSet* used by Algorithm 1. Given a new unlabeled instance, all rules which fire are collected and their associated values are added for each class. After that, the Laplace's accuracy is calculated and the class with the greatest frequency value is selected to label the new instance.

For instance, suppose that given a new instance, three rules R_1, R_2 e R_3 fire for this instance, and the class predicted by R_1 and R_3 is different from the class predicted by R_2. Assuming $l_1r_1 = 30$, $l_1\bar{r}_1 = 5$, $l_2r_2 = 41$, $l_2\bar{r}_2 = 2$, and $l_3r_3 = 10$, $l_3\bar{r}_3 = 0$, as the class predicted by R_1 and R_3 is the same, their associated values are added, given $l_{1,3}r_{1,3} = 40$, $l_{1,3}\bar{r}_{1,3} = 5$. The Laplace's accuracy for both rules R_1 and R_3 is $\texttt{lacc}(R_1, R_3) = (40 + 1)/(40 + 5 + 2) = 0.87$ and for rule R_2 is $\texttt{lacc}(R_2) = (41 + 1)/(41 + 2 + 2) = 0.93$. Therefore, as $\texttt{lacc}(R_2) > \texttt{lacc}(R_1, R_3)$, the class predicted by rule R_2 is selected to label the new instance.

Table 2
Datasets description

Dataset	#Instances	Duplicated or conflicting	#Features (cont.,nom.)	#Classes	Majority error (%)	Missing values
bupa	345	4	6 (6,0)	2	42.03	N
pima	769	1	8 (8,0)	2	34.98	N
breast-cancer	699	8	10 (10,0)	2	34.48	Y
hungaria	294	1	13 (13,0)	2	36.05	Y
crx	690	0	15 (6,9)	2	44.49	Y
hepatitis	155	0	19 (6,13)	2	20.65	Y
anneal	898	12	38 (6,32)	5	23.83	Y
sonar	208	0	60 (60,0)	2	46.63	N
genetics	3190	185	60 (0,60)	3	48.12	N
dna	3186	185	180 (0,180)	3	48.09	N

4. Experimental setup

In order to evaluate the XRULER system, several experiments were conducted on 10 real world domains. Most of datasets are from the UCI Irvine Repository [4] except for dna and genetics, which are from the StatLog project [12].

Two inducers, C4.5 [16] and CN2 [6], have been used in this work. These inducers are well known in the ML community and belong to the eager learning approach. In this approach, the algorithms greedily compile the training data into an intentional concept description, such as a rule set or decision tree, discarding the data after this process. Only the learned concept is used to classify new cases. C4.5 induces decision trees, while CN2 induces production rules.

It is important to observe that each dataset has not been pre-processed in any way, for example, by removing or replacing missing values or transforming features. Furthermore, each individual inducer was run with default options setting for all parameters, i.e. no attempt was made to tune any inducer. We have used both the Laplace's accuracy (`lacc`) and novelty (`nov`) as criteria to select the best rule.

Table 2 summarizes the datasets used in this work. It shows, for each dataset, the number of instances (#Instances), number and percentage of duplicate (appearing more than once) or conflicting (same attribute-value but different class) instances, number of features (#Features) continuous and nominal, class distribution, the majority error and if the dataset have at least one missing value. Datasets are presented in ascending order of the number of features, as will be in the remaining tables and graphs. The experiments were performed in the following way:

1. Each original dataset was randomly divided into two disjoint subsets: the learning subset containing 70% of the instances and the evaluation subset containing the remaining 30%. The evaluation subset was never used by XRULER during the learning step.
2. The learning subset was then sampled 10 times using *bootstrap*, resulting in 10 training samples and 10 test samples.
3. For each training *bootstrap* sample, we run the inducers C4.5 and CN2, thus yielding 20 classifiers (10 induced by C4.5 and 10 induced by CN2).
4. After that, the cover algorithm was applied in order to extract the minimal rule set covering the instances belonging to the learning subset. The cover algorithm was applied twice: using the Laplace's accuracy, as well as using novelty.
5. Finally, the minimal rule set accuracy was evaluated using the instances contained in the evaluation subset. As a baseline, the inducers C4.5 and CN2 were applied to the learning subset and their accuracy was measured on the evaluation subset.

Table 3
Accuracy of C4.5, CN2, and XRULER using bootstrap

Dataset	C4.5	CN2	XRULER (`lacc`)	XRULER (`nov`)
bupa	64.07 ± 1.05	65.43 ± 1.02	66.12 ± 1.78	64.56 ± 2.00
pima	74.44 ± 0.83	72.74 ± 0.88	73.17 ± 0.76	73.74 ± 0.49
breast-cancer	93.82 ± 0.30	95.07 ± 0.32	94.31 ± 0.37	93.97 ± 0.57
hungaria	77.62 ± 1.55	78.18 ± 1.38	77.05 ± 1.52	78.98 ± 1.35
crx	85.91 ± 0.72	82.71 ± 0.45	84.40 ± 0.69	85.60 ± 0.66
hepatitis	76.74 ± 1.58	79.35 ± 0.81	79.78 ± 1.38	77.61 ± 1.65
anneal	90.65 ± 0.70	91.82 ± 0.99	96.77 ± 0.51	93.83 ± 1.51
sonar	72.42 ± 1.45	71.29 ± 1.66	74.19 ± 2.07	71.45 ± 1.25
genetics	93.60 ± 0.27	79.85 ± 1.59	93.00 ± 0.15	89.85 ± 1.67
dna	92.49 ± 0.34	87.94 ± 0.27	92.45 ± 0.31	93.16 ± 0.21
Average	82.18	80.44	83.12	82.28

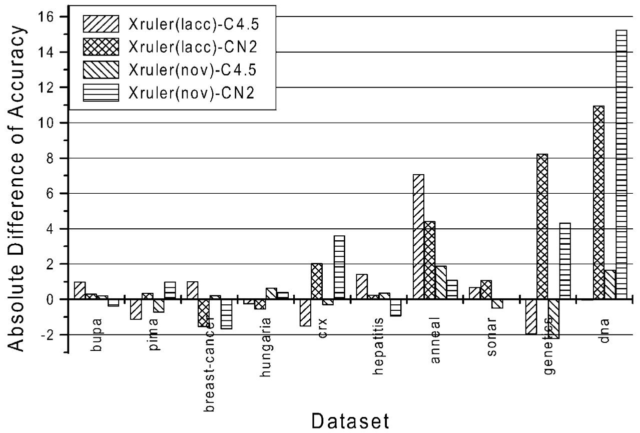

Fig. 2. Absolute difference in standard deviations of accuracy.

6. Since the datasets have moderated size, all these steps were repeated 10 times to provide better estimates. However, for large datasets typically found in DM tasks, this would not be necessary.

5. Results

Table 3 shows the accuracy (mean and standard deviation) obtained by C4.5, CN2, and XRULER using the Laplace's accuracy (`lacc`), as well as novelty (`nov`). To determine whether the difference between them, i.e. XRULER (using `lacc` and `nov`) against the standard inducers C4.5 and CN2, is significant or not, we show in Fig. 2 one graph with four consecutive bars for every dataset. Each bar corresponds to the mean accuracy divided by the standard deviation. The variance of the significance test is the average variance of both algorithms and a standard normal distribution is assumed. Any bar having length greater than two indicates that this result is significant at the 95% confidence level. When the bar is above zero, it means that the first algorithm (XRULER) outperforms the second (C4.5 or CN2); if the bar is below zero, then the second algorithm outperforms the first. Whenever the bar length is

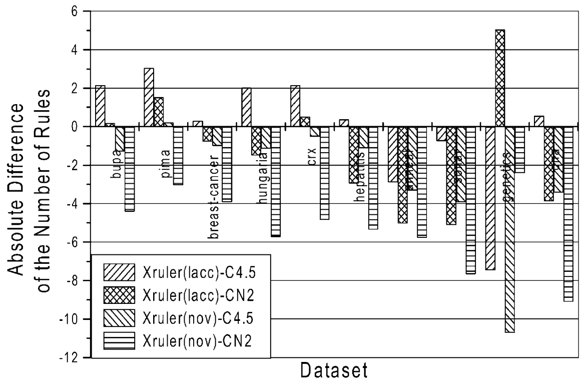

Fig. 3. Absolute difference in standard deviations of the number of rules.

above (below) two, it means that the first (second) algorithm significantly outperforms the second (first) algorithm.

Table 4 shows the average number of rules and Fig. 3 shows the absolute difference (in standard deviations) of the number of rules. Negative (positive) values indicate that XRULER reduced (increased) the number of rules, when compared with the number of rules induced by C4.5 and CN2. As can be noted, XRULER seems to increase the number of rules in datasets with small number of features and instances but the number of rules decreases in datasets with larger dimensions, except for genetics using CN2.

Table 5 provides a comparison between accuracy (horizontal axis) and the number of rules (vertical axis) among XRULER (indicated by x) and C4.5 or CN2. The first quadrant, indicated by ($\oplus$, $\oplus$), shows the cases where an increment in the accuracy, as well as in the number of rules were obtained. The second quadrant, indicated by ($\ominus$, $\oplus$), shows the cases, where a decrease in the accuracy but an increment in the number of rules were obtained. The third quadrant, indicated by ($\ominus$, $\ominus$), shows the cases, where a decrease both in the accuracy and in the number of rules were obtained. The fourth quadrant, indicated by ($\oplus$, $\ominus$), shown the cases, where an increment in the accuracy but a decrease in the number of rules were obtained. The symbol '•' indicates that the corresponding difference both for the accuracy and the number of rules is significant at the 95%

Table 4
Number of rules for C4.5, CN2, and XRULER

Dataset	C4.5	CN2	XRULER (`lacc`)	XRULER (`nov`)
bupa	18.40 ± 4.38	27.60 ± 2.01	28.00 ± 0.94	11.60 ± 3.03
pima	19.20 ± 9.34	39.80 ± 5.37	49.30 ± 3.16	21.40 ± 2.80
breast-cancer	11.20 ± 4.10	14.80 ± 1.32	12.60 ± 2.50	6.90 ± 1.52
hungaria	10.00 ± 2.98	21.70 ± 2.45	17.10 ± 1.85	6.40 ± 1.07
crx	16.20 ± 7.63	31.70 ± 2.67	33.90 ± 3.18	12.00 ± 3.09
hepatitis	6.50 ± 2.37	15.10 ± 1.91	7.60 ± 1.71	3.70 ± 0.95
anneal	45.80 ± 11.10	45.90 ± 6.30	13.80 ± 1.03	8.90 ± 1.29
sonar	12.60 ± 1.58	22.50 ± 1.96	11.20 ± 1.03	4.70 ± 1.25
genetics	276.10 ± 20.62	65.20 ± 9.64	118.80 ± 4.44	32.40 ± 9.61
dna	70.00 ± 9.49	99.10 ± 5.76	75.50 ± 1.96	34.60 ± 4.14
Average	48.60	38.34	36.78	14.26

Table 5
Accuracy versus number of rules

Dataset	(⊖, ⊕)				(⊕, ⊕)			
	x(lacc) −C4.5	x(lacc) −CN2	x(nov) −C4.5	x(nov) −CN2	x(lacc) −C4.5	x(lacc) −CN2	x(nov) −C4.5	x(nov) −CN2
bupa	○					○		
pima	○	○	○					
breast-cancer					○			
hungaria	○							
crx	○					○		
hepatitis					○			
anneal								
sonar								
genetics						•		
dna	○							
bupa			○	○				
pima				○				
breast-cancer		○	○	○				
hungaria		○	○					○
crx							○	•
hepatitis		○	○	•				
anneal					•	•	○	○
sonar				○	○	○	○	
genetics	○		•					
dna						•	○	•
	(⊖, ⊖)				(⊕, ⊖)			

confidence level; ‘○’ indicates that the difference is not significant at the 95% level. It can be seen that the results are more concentrated on the third (⊖, ⊖) and fourth (⊕, ⊖) quadrants. On the third quadrant, there is a concentration of results for small datasets, except for genetics dataset when comparing XRULER (nov) with C4.5, where there is a significant reduction both in the accuracy and in the number of rules. On the other hand, on the fourth quadrant, it is possible to note a concentration of results for large datasets. Considering all quadrants, it is possible to note that the majority of significant results are concentrated on the fourth quadrant, where an increment in accuracy and reduction in the number of rules were obtained by XRULER. These results show that XRULER induces less rules than the standard inducers (C4.5 and CN2) and those rules classify instances with greater accuracy than the standard classifiers.

6. Conclusion

In this work, we described a system under development, which addresses the problem of predictive DM, whenever the result of learning can be expressed in the form of classification rules. Our framework assumes that the dataset is a normal database attribute-value table, which consists of n records described by m distinct features that have been classified into k known classes.

There are many different methods for constructing classification rules although none of them is universally the best, since different application domains lead to different problems, requiring different solutions. This being the case, it is expected that a combination of these methods, as proposed in this work, may yield better classification rules. Such combinations can be made in various ways. We consider simplicity of rules as a goal in itself, although simplicity does not necessarily lead to greater accuracy.

We performed experiments with the XRULER system in 10 datasets using the C4.5 and CN2 inducers. The cover algorithm used two criteria to select the best rule: Laplace’s accuracy and novelty. On average, XRULER showed greater accuracy than the standard inducers C4.5 and CN2. The Laplace’s accuracy presented greater increase in accuracy than novelty.

Another advantage obtained by XRULER is related to the number of rules obtained by the cover algorithm. On average, there was a decrease in the number of rules using the Laplace’s accuracy; for novelty the decrease was more than a half (above 60%). This was accomplished by a decrease in the error rate above 0.5% for C4.5 and almost 10% for CN2. In experiments reported in Ref. [17], where the original CN2 inducer was modified in order to consider novelty for selecting the best rule, on average, the novelty provided a reduction of nine times the number of rules induced but with an increase in the error rate of about 5%. Thus, the results presented by XRULER show that it is possible to combine symbolic classifiers into a final symbolic classifier decreasing the number of rules without decreasing the accuracy.

References

[1] J.A. Baranauskas, M.C. Monard, An environment for rule extraction and evaluation from databases, in: M.C. Monard, J.S. Sichman (Eds.), Proceedings of the IBERAMIA/SBIA, Atibaia, SP, Brazil, 2000, pp. 187–196.

[2] J.A. Baranauskas, M.C. Monard, G.E.A.P.A. Batista, A computational environment for extracting rules from databases, in: N. Ebecken, C.A. Brebbia (Eds.), Proceedings of the Second International Conference on Data Mining, Cambridge, UK, 2000, pp. 321–330.

[3] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: bagging, boosting, and variants, Machine Learning 36 (1999) 105–139.

[4] C.L. Blake, C.J. Merz, UCI repository of machine learning databases, 1998, http://www.ics.uci.edu/~mlearn/MLRepository.html.

[5] L. Breiman, Bias, variance and arcing classifiers, Technical Report 460, Statistics Department, University of California, 1996, ftp://ftp.stat.berkeley.edu/pub/users/breiman/.

[6] P. Clark, R. Boswell, Rule induction with CN2: some recent improvements, in: Y. Kodratoff (Ed.), Proceedings of the Fifth European Conference (EWSL 91), Springer, Berlin, 1991, pp. 151–163.

[7] T.G. Dietterich, E.B. Kong, Machine learning bias, statistical bias, and statistical variance of decision tree algorithms, 1997, ftp://ftp.cs.orst.edu/pub/tgd/papers.

[8] U.M. Fayyad, S.G. Djorgovski, N. Weir, From digitized images to on-line catalogs: Data mining a sky survey, AI Magazine 17 (2) (1996) 51–66.

[9] Y. Freund, R.E. Schapire, Experiments with a new boosting algorithm, Proceedings of the 13th International Conference on Machine Learning, Lake Tahoe, California, Morgan Kaufmann, San Francisco, CA, 1996pp. 123–140.

[10] N. Lavrač, P. Flach, R. Zupan, Rule evaluation measures: a unifying view, in: S. Dzeroski, P. Flach (Eds.), Proceedings of the Ninth International Workshop on Inductive Logic Programming (ILP-99), Lecture Notes in Artificial Intelligence, vol. 1634, Springer, Berlin, 1999, pp. 74–185.

[11] D.D. Margineantu, T.G. Dietterich, Pruning adaptive boosting, Proceedings of the 14th International Conference on Machine Learning, San Francisco, Morgan Kaufmann, San Francisco, CA, 1997pp. 211–218.

[12] D. Michie, D.J. Spiegelhalter, C.C. Taylor (Eds.), Machine Learning, Neural and Statistical Classification, Ellis Horwood, Chichester, UK, 1994.

[13] T.M. Mitchell, Machine Learning, McGraw-Hill, New York, 1998.

[14] MySQL, The MYSQL server, 2000, http://www.mysql.com.

[15] R.C. Prati, J.A. Baranauskas, M.C. Monard, Extracting information for evaluating rules induced by Machine Learning algorithms (in Portuguese), Technical Report 145, ICMC-USP, 2001, ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_145.ps.zip.

[16] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Francisco, CA, 1993.

[17] L. Todorovski, P. Flach, N. Lavrač, Predictive performance of weighted relative accuracy, in: D.A. Zighed, J. Komorowski, J. Zytkow (Eds.), Fourth European Conference on Principles of Data Mining and Knowledge Discovery (PKDD2000), Springer, Berlin, 2000, pp. 255–264.

[18] L. Wall, T. Christiansen, R.L. Schwartz, Programming Perl, O'Reilly & Associates, Inc, 1996.